# Évaluation et Comparaison de Métriques de Robustesse pour l'Ordonnancement de Graphes de Tâches sur des Systèmes Hétérogènes

Louis-Claude Canon

**RenPar'18 / SympA'2008 / CFSE'6**
**Fribourg, Suisse, du 11 au 13 février 2008**

# Évaluation et Comparaison de Métriques de Robustesse pour l'Ordonnancement de Graphes de Tâches sur des Systèmes Hétérogènes

Louis-Claude Canon

LORIA, Nancy-Université
Campus Scientifique – BP 239
54506 Vandœuvre-lès-Nancy Cedex, France
louis-claude.canon@loria.fr

---

**Résumé**
Un ordonnancement est dit robuste s'il est capable d'absorber des variations dans les durées des tâches tout en maintenant une solution stable. Cette notion intuitive de la robustesse a induit beaucoup d'interprétations et de métriques différentes. Cependant, il n'existe pas de comparaison entre ces dernières. Nous comparons d'abord différentes méthodes d'évaluation de ces métriques et présentons ensuite une étude statistique de façon à montrer comment elles sont corrélées dans le cadre de l'ordonnancement de graphes de tâches.

**Mots-clés :** Ordonnancement ; Graphe de Tâches ; Stochastique ; Robustesse ; Statistiques

---

## 1. Introduction

L'idée qui sous-tend le calcul distribué consiste à mutualiser la puissance de calcul d'une quantité conséquente d'ordinateurs interconnectés par un réseau (Internet, ...), dans le but de rendre efficace le traitement de problème de grande taille. C'est un sujet qui entre dans le cadre plus général du calcul haute performance, celui-ci constituant le support de nombreux outils industriel, ainsi que de sciences faisant usage de simulation.
Dans notre optique, nous considérons une application unique qui s'exécute de façon distribuée et qui se décompose en un ensemble de tâches liées par des contraintes de précédence. L'ordonnancement intervient lorsque l'on souhaite définir à quel moment et sur quelle ressource une tâche donnée doit s'exécuter. Plusieurs métriques permettent d'évaluer le résultat de cette étape en fonction des critères fixés (utilisation des ressources, temps total, fiabilité, ...).
La robustesse est un critère à prendre en compte dés lors que l'on considère les incertitudes inhérentes à tout système de calcul. Dans une grille classique, les temps de communications varient souvent en fonction de l'état du réseau, de présence de congestion, de contention et de nombreux autres paramètres. Les temps de calcul sur les machine ne sont eux-même pas fixes, les systèmes d'exploitation pouvant préempter les tâches soumises pour leur propre fonctionnement. Il s'agit donc d'une problématique fondamentale pour l'ordonnancement sur des plateformes à large échelle et on s'efforcera donc de produire des ordonnancements peu sensibles à ces variations.
Avant de construire une heuristique, il est nécessaire d'étudier de manière approfondie les métriques à optimiser. Celles-ci sont en effet multiples et seront présentées dans la section 3.

## 2. Ordonnancement stochastique

### 2.1. Ordonnancement traditionnel

L'ordonnancement est un thème commun à de nombreuses communautés (recherche opérationnelle, temps réel, parallélisme, planification, IA ...) et regroupe donc beaucoup de notions. Nous nous limitons dans notre cas à l'ordonnancement d'une application parallèle sur une plateforme distribuée hétérogène. L'application est modélisée par un graphe orienté acyclique $G = (V, E, C)$, où $V$ est l'ensemble des nœuds représentant des tâches et $E$ est l'ensemble des arcs qui définissent les dépendances entre tâches. $C$ est

l'ensemble des volumes de communication entre 2 tâches liées par une dépendance. Cette définition équivaut à définir un ordre partiel sur l'ensemble des tâches de l'application parallèle en incluant des communications pour chacune des dépendances.
La plateforme cible de l'ordonnancement se compose d'un ensemble de $m$ ressources hétérogènes, c'est à dire que les temps de calcul des tâches et des communications varient en fonction des ressources concernées. Nous disposons de deux matrices pour caractériser les communications, $A = (\alpha_{i,j})_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant m}$ et $B = (\beta_{i,j})_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant m}$. Le temps pris pour réaliser la communication entre la tâche $v$ sur la ressource $i$ et la tâche $w$ sur la ressource $j$ est donné par $\beta_{i,j} + c_{v,w} \times \alpha_{i,j}$, avec $c_{v,w} \in C$. Pour déterminer les temps de calcul de chaque tâche, nous utilisons le modèle *unrelated* car il est le plus général. Le temps d'exécution d'une tâche est donc différent pour chaque ressource et ne dépend pas de leur vitesse. Il est ainsi possible qu'une ressource donnée soit la plus rapide pour une tâche spécifique, mais moins performante pour les autres.
L'ordonnancement est l'opération qui assigne à chaque tâche $v \in V$ une ressource $i \in [1..m]$ ainsi qu'une date de début $t_v^{déb}$ et une date de fin $t_v^{fin}$. Nous considérerons uniquement des ordonnancements dans lesquels aucun temps mort n'est rajouté artificiellement (nous reviendrons sur cette aspect dans la section 3.3). Il s'agit d'un type d'ordonnancement plus général que ceux produits par les heuristiques de liste, en ce qu'il suffit qu'à chaque moment au moins une tâche appartenant au chemin critique soit exécutée.
À un ordonnancement donné correspond un grand nombre de caractéristiques et donc de mesures possibles. Cela est la conséquence directe de la nature à la fois combinatoire et temporelle du problème. Comme objectifs généralement pris en compte, nous pouvons citer le makespan. Il s'agit du temps total d'exécution de l'application et est défini par $\max_{v \in V} t_v^{fin}$. Il s'agit d'un critère très commun et il constituera le point central de l'étude qui suit.

### 2.2. Modèle stochastique

Nous présentons dans cette section la façon dont sont modélisées les perturbations ayant lieu sur les durées considérées (temps de calcul et de communication). Nous avons recours à l'utilisation de variables aléatoires (*va*) car elles permettent de modéliser des durées variables suivant une loi de probabilité. Beaucoup de schémas peuvent être considérés : une durée qui suit une loi normale ; l'utilisation d'une loi construite empiriquement ; l'absence de perturbation (impliquant l'utilisation de la fonction $\delta$ de Dirac) ; ou encore, des retards fixes ayant chacun une probabilité (avec plusieurs fonctions $\delta$ de Dirac pondérées).

#### 2.2.1. Notations et règles de calcul

Nous précisons que les *va* se notent conventionnellement avec des majuscules et que la fonction de densité de probabilité d'une *va* $X$ se note $f_X(x)$ et sa fonction de répartition $F_X(x) = Pr\{X \leqslant x\} = \int_{-\infty}^{x} f_X(t)dt$. Quelques mises en garde nous semblent nécessaires à propos de la manipulation des *va*. Les règles de calcul usuelles ne sont pas valides (les dépendances entre les *va* complexifient les calculs) et les opérations classiques telles que la somme ou le maximum nécessitent des calculs numériques non-triviaux. Nous y reviendront dans la section 4.

#### 2.2.2. Ordonnancement stochastique

Nous partons du schéma déterministe (à savoir le graphe de tâches et les caractéristiques de la plateforme) et modélisons l'incertitude en remplaçant les valeurs déterministes du graphe de tâches par des *va*. Dans notre cas, nous avons considéré uniquement un schéma classique où les *va* suivent une loi de probabilité normale, Gamma, Beta ou Weibull. Cela est conforme avec les observations que nous avons réalisées sur les variations temporelles d'une tâche de nature purement calculatoire.
On définit d'autre part le temps minimal et maximal d'une tâche (même pour une loi normale, les valeurs incohérentes étant dans ce cas numériquement tronquées). Le temps minimal est directement obtenu par la valeur déterministe du graphe de tâches. L'amplitude maximale des valeurs est fixée par le degré d'incertitude qui est général sur tout le graphe. Ce degré est noté UL et représente le ratio entre la valeur maximale et celle minimale.
Le makespan s'évalue à partir des *va* du graphes et est lui-même une *va*. Nous chercherons donc à caractériser sa fonction de répartition. Il a été montré dans [5] que ce problème était en fait #P-Complet,

ce qui correspond à la classe équivalente des problèmes NP-Difficile pour les problèmes de fonction. Nous aborderons plusieurs méthodes d'évaluation dans la section 4.
Dans le cas stochastique, l'ordonnancement ne possède plus la même signification car il n'est plus possible d'assigner des dates de début et de fin aux tâches de façon déterministe. Une stratégie similaire est cependant appliquée, à savoir que chaque ressource exécutera les tâches qui lui sont assignées dans l'ordre donné et ce dès que possible (nous qualifions alors cette étape de *stratégie d'ordonnancement*).

## 3. Métriques de robustesse

La robustesse d'un système se définit usuellement par sa capacité à maintenir son comportement étant donné un certain nombre d'aléas. Pour l'ordonnancement d'une application, il s'agit d'obtenir un déroulement semblable à chacune de ses exécutions. Plus spécifiquement, la distribution du makespan déterminera la répartition des valeurs réelles qui pourront être concrètement rencontrées. Dans notre optique de robustesse, nous souhaitons que cette répartition soit la moins dispersée possible et donc que la densité de probabilité du makespan soit la plus étroite possible.
Il existe un grand nombre de métriques permettant de mesurer la robustesse. Cependant, aucune ne fait l'unanimité dans la littérature et il semble nécessaire, avant même de concevoir des heuristiques d'ordonnancement, de les étudier afin de caractériser ce qui définit le mieux la robustesse dans le cadre de l'ordonnancement. Nous les détaillons successivement ici, en considérant $f(x)$ la fonction de densité du makespan et $\bar{f}$ sa moyenne.

### 3.1. Écart-type

Intuitivement, l'écart-type dénote l'étroitesse de la densité de probabilité. Nous cherchons une valeur minimale de sorte qu'un même ordonnancement donne des valeurs de makespan les plus proches les unes des autres quelles que soient les valeurs que prennent chaque durée. On le formalise ainsi :

$$\sigma(M) = \sqrt{\int_{-\infty}^{\infty} (f(x) - \bar{f})^2 dx}$$

### 3.2. Entropie différentielle du makespan

L'entropie différentielle d'une densité de probabilité [1] caractérise l'incertitude liée à cette fonction. Plus cette incertitude est faible, meilleure sera la robustesse qui caractérise la certitude liée au résultat final. Le cas extrême est donné par l'entropie différentielle de la fonction $\delta$ de Dirac qui vaut $-\infty$ (ce type d'entropie peut effectivement être négative). Formellement, il s'agit de :

$$h(M) = -\int_{-\infty}^{\infty} f(x) \log f(x) dx$$

### 3.3. Slack total moyen

Cette mesure dénote, intuitivement, la quantité de temps pendant laquelle une tâche donnée peut être en retard sans perturber le déroulement initial de l'ordonnancement. Il faut bien comprendre pour cela, que les contraintes de dépendance induiront des périodes d'inactivités sur les processeurs et il y aura alors des vides dans l'ordonnancement.
Le slack [1] est défini de la façon suivante :

$$s_v = M - Bl(v) - Tl(v) \tag{1}$$

où $v \in V$ et $M$ est le makespan de l'ordonnancement évalué. $Bl(v)$ et $Tl(v)$ sont respectivement le *bottom level* et le *top level* que l'on défini de la manière suivante :

$$Bl(v) = comp_v + \max_{w \in Succ(v)} (Bl(w) + comm_{v,w})$$

$$Tl(v) = \max_{w \in Prec(v)} (Tl(w) + comp_w + comm_{w,v})$$

où $comp_v$ est le temps de calcul de la tâche $v$ et $comm_{v,w}$ le temps pris par la communication entre les tâches $v$ et $w$. Les opérateurs $Succ(v)$ et $Prec(v)$ donnent respectivement les successeurs et les prédécesseurs de $v$. Ces définitions caractérisent la longueur du chemin d'une tâche $v$ à la tâche initiale (pour le top level) ou à la tâche finale (pour le bottom level).
Nous calculons le slack moyen de chaque tâche, puis nous sommons ces valeurs. L'idée consiste à considérer qu'une tâche ayant beaucoup de slack pourra absorber certains retards sans altérer le makespan. Comme mentionné précédemment, aucun temps mort n'est rajouté (contrairement à [2]) et le slack mesuré correspond uniquement à celui naturellement présent dans l'ordonnancement. Nous considérons que le problème de l'ajout de slack "artificiel" peut être traité séparément.

### 3.4. Écart type des slacks moyennes

Cette fois-ci, au lieu de prendre en compte la somme des slacks, nous considérons leurs écarts-types. En effet, nous ne souhaitons pas que seules quelques unes des tâches puissent absorber beaucoup d'incertitude, mais que le plus grand nombre puissent en absorber. Ainsi, plus cet écart-type est faible, plus les différentes slacks sont proches de la moyenne.

### 3.5. Retard moyen

Il s'agit du retard moyen d'un ordonnancement par rapport à sa moyenne qui est défini dans [10]. Si cette métrique est large, l'ordonnancement est de fait peu robuste. Nous le formalisons ainsi :

$$R = \int_{\bar{f}}^{\infty} (x - \bar{f}) \frac{f(x)}{\int_{\bar{f}}^{\infty} f(x)dx} dx$$

### 3.6. Métrique probabilistique

Une autre manière de considérer la robustesse est de donner un intervalle de valeurs possibles et de maximiser la probabilité d'obtenir un résultat dans cet intervalle. C'est l'approche préconisée par [9]. Nous donnons soit une largeur d'intervalle absolue de taille $2\beta$, c'est à dire indépendante de la moyenne du makespan, soit relative de facteur $\gamma$. Voici les deux expressions :

$$A(\beta) = \int_{\bar{f}-\beta}^{\bar{f}+\beta} f(x)dx \quad \text{et} \quad R(\gamma) = \int_{\frac{\bar{f}}{\gamma}}^{\gamma\bar{f}} f(x)dx$$

Ce bref état de l'art montre la multitude des métriques existantes et justifie donc une comparaison expérimentale de façon à valider celles qui sont les plus pertinentes.

## 4. Évaluation du makespan stochastique

Afin de procéder aux mesures de ces métriques sur les ordonnancements, il est nécessaire d'évaluer une fonction de densité de probabilité du makespan. De nombreuses méthodes ont été proposées dans la littérature et nous détaillons celle qui fut sélectionnée.

### 4.1. Travaux existants

Les quelques travaux en informatique parallèle à ce sujet n'ont pas apportés d'avancées significatives dans notre optique d'évaluation précise du makespan. Certains donnent des bornes plus ou moins spécifiques en ne considérant que des informations partielles. Dans [7], il est démontré que supposer l'indépendance peut être valide dans certains cas grâce à l'approximation d'indépendance de Kleinrock et les conditions pour caractériser ces cas sont données.
Il s'avère, d'autre part, qu'il s'agit d'un problème bien connu de la communauté de la recherche opérationnelle. Le vocabulaire et les notations utilisés diffèrent cependant (réseau PERT ou d'activités en lieu et place de graphe de tâches). Nous pouvons classifier l'ensemble des travaux liés en quatre catégories :

**Calcul exact** Bien que le problème reste en toute généralité #P-Complet, il est possible pour certain type de graphe de déterminer de façon exact le résultat efficacement. C'est le cas des graphes séries-parallèles qui peuvent se réduire successivement en un seul nœud dont la *va* est le makespan. Dans ce dernier cas, l'algorithme est en temps polynômial.

**Bornes exactes** La fonction de densité de probabilité peut être bornée supérieurement et inférieurement par un certain nombre de techniques. Supposer l'indépendance lors du calcul des maximums permet d'obtenir une borne supérieure. Une autre technique consiste à transformer le graphe en graphe série-parallèle puis à appliquer la méthode exacte précédente, ce qui affine la borne supérieure par rapport à la technique précédente. Celle-ci a été développée par Dodin [3].

**Approximation** Certaines méthodes ne permettent pas d'assurer une borne mais tentent soit d'en approximer une, soit d'approximer le résultat. Dans l'algorithme de Spelde [8] par exemple, seul les chemins les plus critiques sont considérés. Ils sont alors supposés être complètement indépendants et les *va* sont approximées en lois normales.

**Simulation de Monte Carlo** L'idée consiste à générer une valeur pour chaque *va* conformément à leur loi de probabilité et à évaluer le makespan. L'opération est répétée un grand nombre de fois et la fonction de densité de probabilité du makespan peut ainsi être construite.

### 4.2. Supposition de l'indépendance

Dans un premier temps, nous avons supposé l'indépendance entre les *va* pour les maximums de façon à obtenir une borne supérieure et nous avons utilisé ce résultat comme approximation. Celle-ci a été validée en la comparant avec le résultat donné par une méthode de simulation de Monte Carlo basique.
Lors de ce calcul, les deux opérations à effectuer sont la somme et le maximum, la somme lorsqu'un nœud n'a qu'un unique prédécesseur (il faut alors rajouter le temps de calcul ou de communication au temps déjà considéré) et le maximum quand la tâche a plusieurs prédécesseurs (elle doit donc attendre plusieurs tâches et on fait le maximum des temps de fin de toutes ces tâches) suivi d'une somme comme dans le cas d'une seul prédécesseur.

#### 4.2.1. Somme de variables aléatoires indépendantes

Soit $Z = X + Y$, une somme de *va* indépendantes qui se calcule en réalisant la convolution des fonctions de densité de probabilité de X et Y. Cela se note $f_Z(z) = (f_X * f_Y)(z)$. Numériquement, ce calcul est en complexité $\Theta(n^2)$ (où $n$ est le nombre d'échantillons numériques) et peut être optimisé en ayant recours à la transformée de Fourier. En effet dans le domaine fréquentiel, la convolution se transforme en produit qui est une opération linéaire. La transformation peut être réalisée par l'algorithme de transformée de Fourier rapide (FFT) qui est en $\Theta(n \log n)$. En plus de cela, le calcul numérique présente quelques écueils. Lorsque deux fonctions sont définies sur des intervalles différents, un rééchantillonage doit précéder la FFT, ce qui peut conduire à un perte de précision conséquente ou une augmentation du nombre d'échantillons considérable. Nous utilisons donc la méthode OverLap-Add (voir [6] pour plus de détails) qui permet de réaliser la convolution de façon fragmentée pour des signaux définis sur des intervalles de tailles différentes et de conserver une complexité et une précision convenable.

#### 4.2.2. Maximum de variables aléatoires indépendantes

Le maximum de deux *va* indépendantes ($Z = \max(X, Y)$) est calculatoirement plus simple à réaliser. Nous l'obtenons ainsi : $F_Z(z) = F_X(z) \times F_Y(z)$. Cela se prouve aisément en réécrivant les fonctions de répartition : $F_Z(z) = \Pr\{Z \leqslant z\} = \Pr\{X \leqslant z \cap Y \leqslant z\} = \Pr\{X \leqslant z\} \times \Pr\{Y \leqslant z\} = F_X(z) \times F_Y(z)$. Comme le passage de la fonction de répartition à la fonction de densité de probabilité se fait par une dérivée et que cela est numériquement délicat, nous préférons obtenir une densité de probabilité et nous dérivons donc la formule pour obtenir : $f_Z(z) = F_X(z) \times f_Y(z) + f_X(z) \times F_Y(z)$.

#### 4.2.3. Implémentation

Un programme de simulation de quelques milliers de lignes de code a été mis au point. La GSL (GNU Scientific Library [4]) a été choisie pour les opérations statistiques et numériques (génération des fonctions de densité de probabilité, interpolation par spline, FFT et lissage). Des méthodes numériques ont aussi été utilisées pour l'intégration (Simpson) et pour optimiser la FFT (méthode OverLap-Add).
En terme de précision, il est notable que la supposition d'indépendance proposée par [7] est largement insuffisante dès lors que la taille des graphes dépasse les 100 nœuds ou que les indices d'incertitude deviennent élevés (pour des UL $> 1.1$). Les résultats obtenus avec les algorithmes de Dodin et de Spelde sont sensiblement identiques à cet égard.

## 5. Comparaison statistique

### 5.1. Démarche expérimentale

Pour comparer ces différentes définitions, nous nous sommes placés dans une situation générale en générant des ordonnancements aléatoires pour des graphes donnés. Grâce aux méthodes d'évaluation de la distribution du makespan présentées dans la section 4, nous avons ensuite mesuré les métriques pour caractériser leurs évolutions réciproques. Ces comparaisons se sont faites par l'utilisation de méthodes statistiques pour étudier la corrélation entre deux variables aléatoires.

Les graphes de tâches utilisés ont été soigneusement sélectionnés afin de représenter un spectre assez large de cas possibles. Nous avons d'un coté des graphes d'applications réelles (décomposition de Cholesky et élimination de Gauss) et des graphes aléatoires. Pour chaque graphe, nous faisons varier le nombre de tâches ($| V |= 10$, 30 et 100) et le degré d'incertitude (UL = 1.01 et 1.1). D'autre part, jusqu'à 10 graphes ont été générés pour chaque taille. Nous avons fixé un certain nombre de paramètres comme le rapport du temps de communication total sur le temps de calcul total (CCR = 0.1) et le coefficient de variation qui définit les temps de calcul de chaque tâche sur chaque ressource (en l'occurrence, ces temps sont distribués suivant un loi Gamma ayant pour moyenne 20 et pour écart-type 10). La forme des graphes aléatoires est prédéterminée pour les graphes réels et générée aléatoirement pour les autres suivant une approche par niveau. Une fois ces graphes obtenus, chaque valeur déterministe a été substituée par une distribution Beta ayant pour paramètres $\sigma = 2$ et $\theta = 5$.

Comme mentionné précédemment, seules des solutions aléatoires ont été prise en compte. La construction d'un ordonnancement s'est faite en itérant le processus suivant :

1. Sélectionner une tâche aléatoire parmi celles qui sont disponibles
2. Assigner cette dernière à un processeur quelconque et la positionner en première position possible
3. Mettre à jour la liste des tâches disponibles

Étant donné les faibles degrés d'incertitude, nous avons sélectionné la méthode d'évaluation de la distribution du makespan qui suppose l'indépendance entre les *va*. Nous avons cependant montré les limites de cette approche lors de sa validation. Nous montrons ainsi que dans le cas UL = 1.1, les résultats pour des graphes de 1000 tâches perdent leurs significations. Nous ne considérerons donc pas les graphes de 1000 tâches.

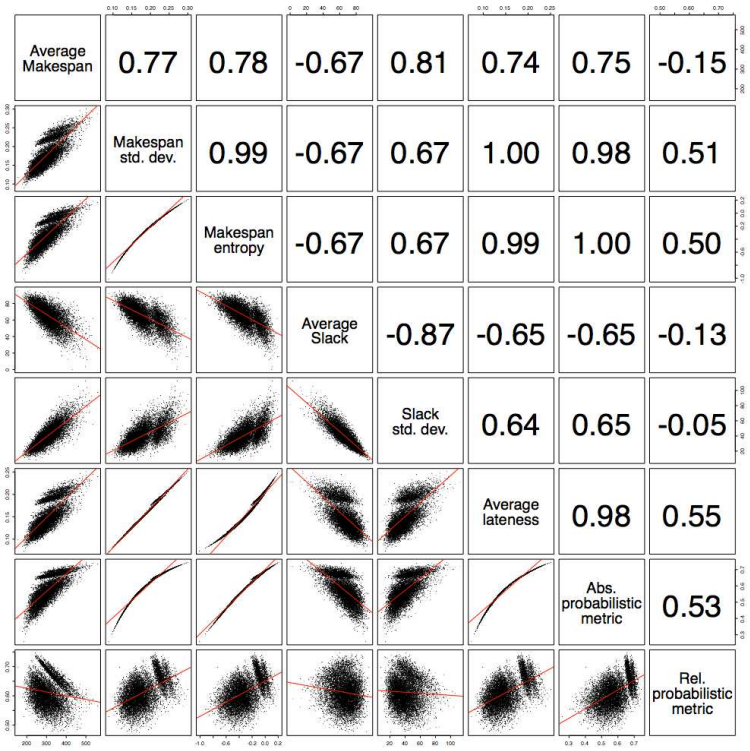


FIG. 1 – Corrélations des métriques pour un graphe aléatoire de 30 tâches sur 8 processeurs avec UL = 1.01. Partie inférieure de la matrice : graphes de 10000 ordonnancements aléatoires. Partie supérieure : valeurs des coefficients de Pearson associés aux ordonnancements aléatoires.

Une fois les graphes obtenus et les ordonnancements générés pour chacuns, les métriques présentées dans la section 3 sont évaluées. De l'ensemble de ces résultats, il est possible de calculer la corrélation entre chaque métrique, ce qui peut se faire par le biais du coefficient de Pearson. Bien qu'il soit limité aux corrélations linéaires, il donne d'excellents résultats dans notre cas. Cette étude statistique est réalisée en utilisant le langage de programmation R, ce dernier nous permettant aussi de générer de façon simple des courbes de corrélation claires.

### 5.2. Résultats obtenus

Sur les 24 résultats générés, l'un des plus représentatifs a été sélectionné (voir la figure 1). Les 8 métriques y sont comparées les unes aux autres, engendrant ainsi une matrice de 64 éléments. Le nom de chaque métrique est donné sur la diagonale, tandis que les ensembles des points accompagnés des régressions linéaires de chaque métrique par rapport aux autres sont tracés sur la partie triangulaire inférieure gauche. Les coefficients de Pearson sont affichés sur la partie restante. Pour faciliter la lecture du graphe, 3 métriques ont été inversées pour que chacun des objectifs soient à minimiser (les points correspondant aux meilleurs résultats seront donc situés dans le coin inférieur gauche des éléments de la matrice). Ces métriques sont le slack total, d'après l'intuition qui y est liée, et les deux métriques probabilistiques. Pour obtenir des détails d'une corrélation donnée, il suffit de partir des deux éléments sélectionnés sur la diagonale et de considérer les éléments qui sont sur la même colonne que l'un et la même ligne que l'autre. D'un coté de la matrice, nous obtenons les points et de l'autre le coefficient de Pearson (une valeur négative signifiant une corrélation négative, c'est à dire que lorsqu'une des métriques augmente, l'autre diminue).

On résume l'ensemble des données obtenues sur la figure 2. Nous réalisons la moyenne des 24 coefficients de Pearson que l'on place au même endroit dans la matrice et dans la partie inférieure gauche, nous plaçons les écarts-types associés. De cette façon, nous voyons rapidement les corrélations significatives (celles dont la valeur absolue est proche de 1 et donc l'écart-type est proche de 0). Par exemple, le retard moyen et la métrique probabilistique absolue sont fortement corrélés (coefficient de Pearson moyen de 0.981 et écart-type de 0.022).

Pearson coefficients (top: mean, bottom: std. dev.)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Average Makespan | 0.767 | 0.762 | -0.385 | 0.537 | 0.756 | 0.734 | -0.467 |
| 0.107 | Makespan std. dev. | 0.996 | -0.460 | 0.480 | 0.999 | 0.982 | 0.148 |
| 0.109 | 0.002 | Makespan entropy | -0.458 | 0.476 | 0.994 | 0.990 | 0.154 |
| 0.407 | 0.301 | 0.299 | Average Slack | -0.873 | -0.461 | -0.444 | -0.134 |
| 0.373 | 0.256 | 0.254 | 0.028 | Slack std. dev. | 0.480 | 0.456 | -0.084 |
| 0.098 | 0.001 | 0.002 | 0.291 | 0.248 | Average lateness | 0.981 | 0.165 |
| 0.104 | 0.021 | 0.029 | 0.299 | 0.256 | 0.022 | Abs. probabilistic metric | 0.184 |
| 0.245 | 0.384 | 0.386 | 0.252 | 0.238 | 0.375 | 0.380 | Rel. probabilistic metric |

FIG. 2 – Moyennes (partie supérieure) et écarts-types (partie inférieure) des coefficients de Pearson sur 24 expériences

### 5.3. Interprétations

#### 5.3.1. Métriques équivalentes

Le premier phénomène remarquable est qu'un certain nombre de métriques sont quasiment équivalentes. L'écart-type du makespan, son entropie différentielle, le retard moyen et la métrique probabiliste absolue sont effectivement fortement corrélés.

Ces corrélations quasi-linéaires indiquent que la forme de la distribution du makespan est similaire quelque soit l'ordonnancement ce qui peut s'expliquer par l'utilisation du théorème de la limite centrale, qui statut que la somme de plusieurs *va* indépendantes tend à être normalement distribuée. Comme de nombreuses sommes sont effectuées pour obtenir le makespan, nous pouvons considérer qu'il s'agit d'une Gaussienne, ce qui justifie ces premières observations.

Ainsi, à l'exception des petits graphes, nous pouvons ne mesurer qu'une seule de ces métriques, la plus simple étant l'écart-type. Nous considérons dans ce cas que l'effet des sommes, lors de l'évaluation du makespan, est prépondérant sur celui des maximums, ce qui se vérifie d'ailleurs expérimentalement.

#### 5.3.2. Antagonisme entre le makespan et le slack total

Si le coefficient de Pearson est plutôt proche de $-0.3$ (ce qui indique un faible degré de corrélation) en ce qui concerne le makespan moyen et le slack total, nous observons que le slack total n'est jamais grand lorsque le makespan moyen l'est et inversement, le makespan moyen n'est jamais petit lorsque le slack total est grand. Nous l'expliquons en considérant que les bons ordonnancements (du point de vue du makespan moyen) n'auront pas beaucoup de temps libre et donc peu de slack. Il existe donc un antagonisme entre ces deux objectifs même s'ils ne sont pas fortement corrélés.

#### 5.3.3. Corrélation entre la moyenne et l'écart-type du makespan

Bien que la corrélation entre la moyenne et l'écart-type du makespan ne soit pas excellente, il existe un fort lien que nous expliquons par la façon dont est modélisée l'incertitude, à savoir par le biais du

paramètre UL. En effet, l'écart-type de chaque *va* est proportionnel à sa moyenne et comme la variance d'une somme de deux *va* est égale à la somme des variances des 2 *va* initiales, l'écart-type du makespan est corrélé à la somme des moyennes des *va* présentes sur le chemin le plus critique – en ignorant l'effet des maximums – et donc à sa moyenne. L'imperfection de la corrélation peut certainement s'expliquer par l'existence des maximums et aussi par les approximations de l'explication.
On peut donc conclure que dans le cas où les degrés d'incertitude sont relatifs et identiques pour chaque *va*, minimiser le makespan moyen permettra aussi d'obtenir de bonnes performances en terme de robustesse.

#### 5.3.4. Indépendance du slack total et des métriques de robustesse

Le résultat le plus surprenant de cette étude est le faible taux de corrélation entre le slack total et l'ensemble des autres métriques robustes. Il semble en effet que maximiser la slack soit un objectif contraire à celui du makespan moyen, lui-même corrélé avec son écart-type. On remarque aussi par le biais d'exemple simple, que l'on peut tout aussi bien avoir des ordonnancements robustes avec beaucoup de slack que des ordonnancements non-robustes possèdant tout autant de slack.

## 6. Conclusion

L'ordonnancement stochastique est encore assez peu étudié dans le cadre du parallélisme et nous pensons qu'il existe un fort potentiel de recherche dans ce domaine, d'autant plus que ces travaux sont à mettre en relation avec les résultats de la recherche opérationnelle particulièrement aboutis.
L'ensemble de cette étude part de la problématique de l'ordonnancement stochastique et du développement de méthodes solides concernant l'optimisation de la robustesse. Nous comparons d'abord différentes méthodes d'évaluation stochastique du makespan et constatons qu'il n'existe pas de méthode idéale dans des situations concrètes. Nous invalidons le critère usuel qu'est le *slack total* et présentons des explications pour cela. Nous réalisons aussi un certain nombre d'observations qui aident à comprendre la structure du problème dans l'objectif de déboucher sur des heuristiques efficaces.

## Bibliographie


1. Bölöni (Ladislau) et Marinesco (Dan C.). – Robust scheduling of metaprograms. *Journal of Scheduling*, vol. 5, n5, septembre 2002, pp. 395–412.
2. Davenport (Andrew J.), Gefflot (Christophe) et Beck (J. Christopher). – Slack-based Techniques for Robust Schedules. *In : Proceedings of the Sixth European Conference on Planning (ECP-2001)*, pp. 7–18. – Toledo, Spain, septembre 2001.
3. Dodin (Bajis). – Bounding the project completion time distribution in PERT networks. *Operations Research*, vol. 33, n4, juillet/août 1985, pp. 862–881.
4. GSL - GNU Scientific Library. – http ://www.gnu.org/software/gsl/.
5. Hagstrom (Jane N.). – Computational complexity of PERT problems. *Networks*, vol. 18, n2, 1998, pp. 139–147.
6. Jr (Thomas G. Stockham). – High-speed convolution and correlation. *In : AFIPS Proc. 1966 Spring Joint Computer Conf*.
7. Li (Yan Alexander) et Antonio (John K.). – Estimating the Execution Time Distribution for a Task Graph in a Heterogeneous Computing System. *In : 6th IEEE Heterogeneous Computing Workshop (HCW'97)*, pp. 172–184. – Geneva, Switzerland, avril 1997.
8. Ludwig (Arfst), Mohring (Rolf H.) et Stork (Frederik). – A Computational Study on Bounding the Makespan Distribution in Stochastic Project Networks. *Annals of Operations Research*, vol. 102, n1–4, février 2001, pp. 49–64.
9. Shestak (Vladimir), Smith (Jay), Siegel (Howard Jay) et Maciejewski (Anthony A.). – A Stochastic Approach to Measuring the Robustness of Resource Allocations in Distributed Systems. *In : Proceedings of the 2006 International Conference on Parallel Processing (ICPP'06)*, pp. 459–470. – Columbus, Ohio, USA, août 2006.
10. Shi (Zhiao), Jeannot (Emmanuel) et Dongarra (Jack J.). – Robust Task Scheduling in Non-Deterministic Heterogeneous Computing Systems. *In : Proceedings of IEEE International Conference on Cluster Computing*. pp. 1–10. – Barcelona, Spain, septembre 2006.